NeRF Is a Valuable Assistant for 3D Gaussian Splatting

Shuangkang Fang[1], I-Chao Shen[2], Takeo Igarashi[2], Yufeng Wang[1], ZeSheng Wang[1],
Yi Yang[3], Wenrui Ding[1], Shuchang Zhou[3]

[1]Beihang University   [2]The University of Tokyo   [3]StepFun

Abstract

We introduce NeRF-GS, a novel framework that jointly optimizes Neural Radiance Fields (NeRF) and 3D Gaussian Splatting (3DGS). This framework leverages the inherent continuous spatial representation of NeRF to mitigate several limitations of 3DGS, including sensitivity to Gaussian initialization, limited spatial awareness, and weak inter-Gaussian correlations, thereby enhancing its performance. In NeRF-GS, we revisit the design of 3DGS and progressively align its spatial features with NeRF, enabling both representations to be optimized within the same scene through shared 3D spatial information. We further address the formal distinctions between the two approaches by optimizing residual vectors for both implicit features and Gaussian positions to enhance the personalized capabilities of 3DGS. Experimental results on benchmark datasets show that NeRF-GS surpasses existing methods and achieves state-of-the-art performance. This outcome confirms that NeRF and 3DGS are complementary rather than competing, offering new insights into hybrid approaches that combine 3DGS and NeRF for efficient 3D scene representation.

1. Introduction

Neural Radiance Fields (NeRF) [41] and 3D Gaussian Splatting (3DGS) [27] have emerged as two prominent methodologies in 3D scene reconstruction, photorealistic rendering, and virtual reality applications [6, 13, 14, 16, 17, 19, 37, 41, 42, 51, 54, 58]. NeRF represents 3D scenes through a continuous volumetric field, capturing intricate details by an MLP-based encoding of color and density at any position in space. However, it requires numerous forward passes through the MLP, limiting its applicability in real-time scenarios. In contrast, 3DGS [27] represents a scene through a set of discrete Gaussians to approximate points in space, which capitalizes on point-based rendering for computational efficiency, providing real-time performance. Nonetheless, the reliance on Gaussian initialization and limited spatial perception can lead to instability in

Figure 1. NeRF-GS establishes a bridge of communication between NeRF and 3DGS, leveraging information sharing, modeling of distinct characteristics, and joint optimization to enable 3DGS to achieve higher fidelity representation. In this case, NeRF-GS outperforms the vanilla 3DGS by 1.8dB in PSNR.

3DGS training [12, 18, 20, 46]. Moreover, the weak correlation between discrete Gaussians results in a lack of smooth spatial transitions [7, 8, 40], which negatively affects the visual quality of the rendered outputs.

To address these deficiencies, existing studies have sought to improve both NeRF and 3DGS. For example, some researches focus on accelerating rendering for NeRF [6, 19, 22, 42, 53, 64], while others improve 3DGS in terms of visual quality [9, 35, 40, 65]. However, most of these studies treat NeRF and 3DGS as independent scene representation paradigms, concentrating on their separate enhancements. Several researchers have attempted to leverage the properties of NeRF to enhance 3DGS, such as initializing 3DGS with NeRF [18, 46], embedding NeRF attributes within 3DGS [36, 40, 46], or creating networks to implicitly estimate 3DGS parameters [8, 35]. However, these approaches primarily focus on individually enhancing variants of 3DGS, without systematically exploring the potential components in the full NeRF pipeline that could benefit 3DGS. They also overlook the modeling of struc-

tural differences between the two, resulting in limited performance gains. Thus, the integration of NeRF and 3DGS methods and the combination of their respective strengths remain underexplored.

To this end, we propose NeRF-GS, a novel framework that integrates the NeRF network into the training of the 3DGS model, leveraging specific NeRF properties to address 3DGS inherent limitations. In revisiting the design space of the 3DGS model, we identify and implement three critical components in the hybrid NeRF-GS framework.

(1) Sharing Mechanism (Sec 4.1): we first introduce a Hash-based network for encoding features in continuous space optimized by NeRF volume rendering, and design strategies to identify potential Gaussian positions. Subsequently, both NeRF and 3DGS share these features to decode additional attributes for their respective spatial points.

(2) Residual Vectors (Sec 4.2): due to inherent differences between the NeRF and 3DGS forms, directly using NeRF-optimized features and NeRF-initialized Gaussian positions does not adequately adapt to the 3DGS branch. To address this, we propose explicitly modeling their discrepancies by optimizing residual vectors for both features and positions to personalize and enhance 3DGS performance.

(3) Joint Optimization (Sec 4.3): we align the attributes and rendering results of spatial points along rays passing through the important Gaussian in the NeRF branch with those in the 3DGS, which reduces feature confusion and ensures mutually beneficial constraints on shared features. Additionally, we leverage NeRF's continuous spatial query capability to assist in adaptive Gaussian growth, achieving efficient joint optimization across different branches.

The hybrid design of NeRF-GS is not a mere combination of NeRF and 3DGS, but rather a systematic and comprehensive integration that considers the interrelations and differences between the two, maximizing the auxiliary role of NeRF in enhancing 3DGS. It is structurally flexible, allowing the 3DGS branch to be independently separated after joint optimization, thus preserving its real-time rendering capability. Experiments conducted on benchmark datasets demonstrate that NeRF-GS significantly outperforms the original 3DGS method in both quantitative and qualitative evaluation. Additionally, the mutual regularization between the dual branches in NeRF-GS notably improves the rendering quality of the 3DGS branch under sparse-view conditions. These findings indicate that these seemingly disparate scene representation methods are, in fact, complementary rather than competitive, providing new insights for exploring further hybrid 3D scene representation techniques.

2. Related work

Implicit Volume Rendering. Implicit methods provide a continuous spatial modeling capability to represent 3D scenes, eliminating the need for discretization [2, 4, 33, 38,

39, 41, 42, 45, 47, 49, 50, 56, 67]. Many methods have been developed on this basis to improve the visual quality and rendering speed. Plenoctrees [64] and Plenoxels [19] render faster than vanilla NeRF by pre-tabulating a Tensor. DeRF [52] and KiloNeRF [53] accelerate speed by partitioning the target scene into smaller MLPs. Instant-NGP [42] introduces a learnable, multi-resolution hash encoding to fit scenes efficiently. Mip-NeRF [4] enhances NeRF with cone tracing multi-scale properties and automatic anti-aliasing. Several methods have demonstrated that the features extracted by NeRF contain significant scene information. For example, Unisurf [48] achieves a detailed mesh by sharing features between NeRF and SDF. DecomNeRF [30] enables semantic-level scene decomposition through feature embedding. PVD [14, 15] facilitates the conversion between different forms of NeRF by distilling features.

Point-based Representations. Recent advances in point-based 3D rendering have shown substantial improvements in rendering efficiency [21, 24–27, 35, 36, 40, 63]. RAIN-GS [26], Agg [60], and NPGs [10] have proposed novel initialization strategies to address the limitations of the initialization from SfM in the original 3DGS. MS3DGS [62], Analytic-Splatting [32], and SA-GS [55] enhance 3DGS performance by introducing strategies to reduce aliasing. Additionally, to mitigate the issues of storage demands in 3DGS, some approaches have achieved lightweight Gaussian representations through parameter compression [3, 34, 43, 44, 66] and pruning [1, 11, 69].

Several studies have explored the complementarity and transfer of characteristics between different 3D representations. Notable examples [18, 46] utilize points extracted from NeRF for 3DGS initialization. VDGS [36] incorporates NeRF concepts by employing implicit MLPs to make 3DGS opacity view-dependent. SplatFields [40] samples implicit features from triplanes, establishing an auto-correlated feature space for estimating Gaussian sphere parameters. Scaffold-GS [35] derives the possible positions and attributes of Gaussian from a set of candidate anchors. Hash-GS [8] and Compact-3DGS [31] leverages NeRF attributes for 3DGS parameter compression. However, these methods mainly adopt certain NeRF-inspired features to independently optimize 3DGS variants, which differ significantly from our methodology. Moreover, these methods typically implement direct transfer of NeRF characteristics to 3DGS without considering their inherent differences, thereby failing to fully exploit the models' potential.

3. Preliminaries

Neural Radiance Fields. NeRF represents scenes using an implicit function that maps spatial points $\mathbf{x} = (x, y, z)$ and view directions $\mathbf{d} = (\theta, \phi)$ to density σ and color $\mathbf{c}$. For a ray originating at $\mathbf{o}$ with direction $\mathbf{d}$, the RGB value
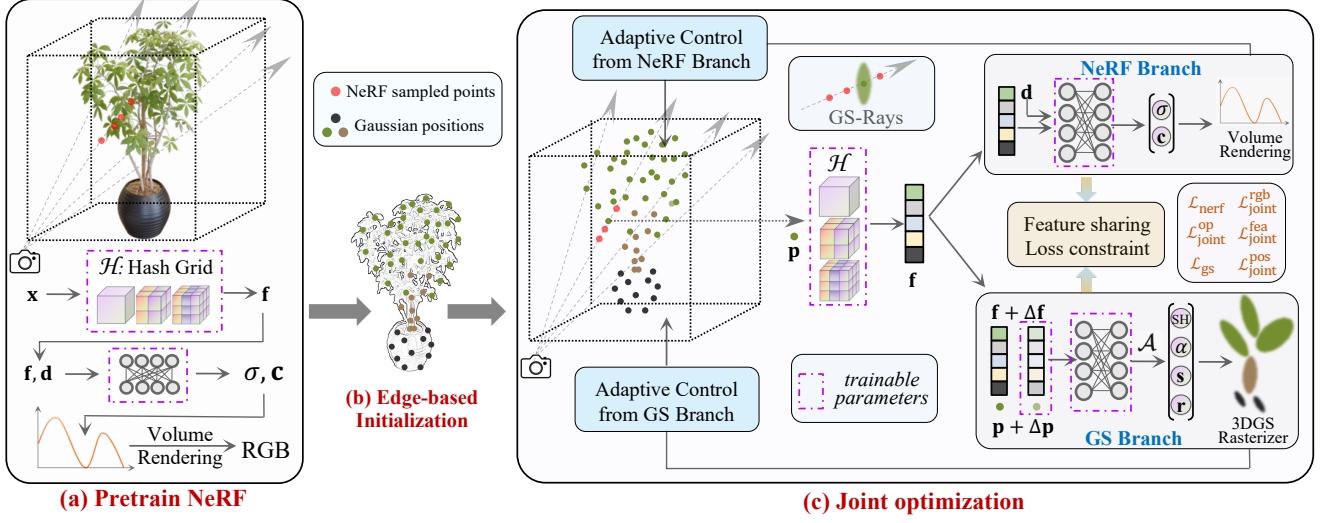
Figure 2. Overview of NeRF-GS. (a) We first pretrain a Hash-based NeRF network to acquire continuous spatial encoding capabilities and implicit scene representation. (b) Utilizing the preliminary scene carved by NeRF, we resample rays corresponding to image edges to obtain potential Gaussian positions, facilitating Gaussian initialization. (c) During joint optimization, the GS branch queries corresponding features **f** from the Hash grid $\mathcal{H}$ for each Gaussian sphere. These features, combined with positions **p** and their respective residual terms ($\Delta\mathbf{f}, \Delta\mathbf{p}$), decode additional Gaussian attributes $\mathcal{A}$, including color, opacity, scale, and rotation vectors. For the NeRF branch, rendering is exclusively performed on rays (GS-Rays) passing through important Gaussian spheres within the view frustum. The two branches are aligned by opacity and RGB values ($\mathcal{L}_{\text{joint}}^{\text{op}}, \mathcal{L}_{\text{joint}}^{\text{rgb}}$), further supervised by reconstruction($\mathcal{L}_{\text{gs}}, \mathcal{L}_{\text{nerf}}$), and residual regularization ($\mathcal{L}_{\text{reg}}^{\text{fea}}, \mathcal{L}_{\text{reg}}^{\text{pos}}$). Simultaneously, we leverage ray attributes from the NeRF branch along with gradient information from the GS branch to achieve adaptive control over Gaussian spheres. The purple dashed box marks the parameters to be trained.

C_{nerf} of the corresponding pixel is computed by numerically integrating the colors $\mathbf{c}_i$ and densities σ_i of the spatial points $\mathbf{x}_i = \mathbf{o} + t_i\mathbf{d}$ sampled along the ray:

$$C_{\text{nerf}} = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))\mathbf{c}_i, \tag{1}$$

where $T_i = \exp(-\sum_{j=1}^{i-1}\sigma_j\delta_j)$ is the accumulated transmittance up to the i-th sample, and δ is the distance between adjacent samples.

Gaussian Splatting. In 3DGS, the scene is represented by a set of anisotropic 3D Gaussian functions, inheriting the EWA volume splatting method [70] and allowing efficient rendering through a tile-based rasterization approach. Typically, 3DGS are initialized from a set of points generated by SfM and can be described by the central position **p** and a covariance matrix Σ that is parameterized using a rotation matrix R and a scaling matrix S as follows:

$$\Sigma = RSS^T R^T. \tag{2}$$

3DGS uses a quaternion **r** to represent rotation and a vector **s** for scaling. Each Gaussian is also associated with an opacity $\alpha \in [0, 1]$ and a set of spherical harmonics (SH) coefficients to define the view-dependent color **c**. By projecting the 3D Gaussian into 2D, the color and opacity coverage of each projected Gaussian are evaluated and the pixel

color C_{gs} can be calculated by sequentially blending all 2D Gaussians that contribute to the pixel, as follows:

$$C_{gs} = \sum_{i \in N} \mathbf{c}_i\alpha_i \prod_{j=1}^{i-1}(1 - \alpha_j). \tag{3}$$

4. NeRF-GS

Our objective is to integrate the full NeRF pipeline into 3DGS model training and utilize specific properties of NeRF to address the limitations of 3DGS. Fig. 2 illustrates an overview of our method. We start by independently training the NeRF branch to model a spatially continuous Hash feature and initialize the Gaussian spheres in the 3DGS branch, enabling spatial awareness and feature sharing between the two branches (Sec 4.1). We design a neural GS branch derived from these shared features. To fill the gap between NeRF and 3DGS representations, we further optimize each Gaussian with a residual feature vector and a position offset vector, using the refined vectors to infer additional Gaussian attributes (Sec 4.2). During joint optimization, we introduce 'GS-Rays', defined as rays connecting important Gaussian centers within the view frustum to the camera origin, serving as query rays for the NeRF branch. Along these GS-Rays, we achieve mutual constraint between NeRF and GS branches by minimizing differences in their spatial attributes and rendering results. Furthermore,

we leverage NeRF to facilitate Gaussian adaptive growth in regions challenging for 3DGS perception (Sec 4.3).

4.1. Sharing Mechanism in Dual-branch

NeRF for Prior Sharing. NeRF represents the scene as a continuous volumetric field, allowing arbitrary queries of spatial points to obtain density σ and color $\mathbf{c}$. This guarantees that every Gaussian has a corresponding NeRF feature, and volume rendering creates strong spatial correlations that address 3DGS's limitations in discrete point representation and weak spatial relationships. To achieve efficient feature sharing, we construct a hash feature extraction network $\mathcal{H}$ that extracts multi-scale features $\mathbf{f}$ from a spatial point $\mathbf{x}$. As in INGP [42], the density σ is derived from the spatial feature $\mathbf{f}$, with the color $\mathbf{c}$ derived by combining $\mathbf{f}$ and the direction vector $\mathbf{d}$, as shown below.

$$\mathbf{f} = \mathcal{H}(\mathbf{x}), \ \ \sigma = F_\sigma(\mathbf{f}), \ \ \mathbf{c} = F_c(\mathbf{f}, \mathbf{d}). \tag{4}$$

Once σ and $\mathbf{c}$ are obtained, the images can be rendered by Eq. 1. After the NeRF branch has been pre-trained, the features $\mathbf{f}$ can be shared with the 3DGS branch to capture similar information at corresponding spatial points.

Edge-based Initalization. Similar to RadSplat [46], we estimate initial Gaussian positions by computing the median ray depth z. However, unlike RadSplat, which uniformly samples one million points from all rays for Gaussian initialization, we assign higher sampling weights to rays corresponding to high-frequency image textures, as these textures define scene contours. Specifically, we apply edge detection to extract image textures, designate their corresponding rays as edge rays, and then estimate the potential Gaussian position set $\mathcal{G}_{init}$ using the following approach:

$$\mathcal{G}_{init} = \{\mathbf{p}_i \mid \mathbf{p}_i \in \{\mathcal{P}_{edge}, \mathcal{P}_{random}\}\}, \tag{5}$$

where $\mathcal{P}_{edge}$ and $\mathcal{P}_{random}$ are points in edge rays and random rays, respectively, and Gaussian position $\mathbf{p}_i = \mathbf{o} + \mathbf{d}_i * z_i$. Our design tightly integrates the NeRF and GS branches, ensuring that the initialized points continue to share spatial information and undergo further joint optimization, which is completely different from RadSplat and NeRF-init [18] that treat initialization as an independent step.

Neural GS Derivation from Shared Features. Unlike the vanilla 3DGS [27], which directly optimizes Gaussian properties, we embed shared features $\mathbf{f}$ into the GS branch. Specifically, we use a tight MLP F_{gs} transforms $\mathbf{f}$ and $\mathbf{p}$ into Gaussian attributes $\mathcal{A}$ as follows:

$$\mathcal{A} = F_{gs}(\mathbf{p}, \mathbf{f}), \tag{6}$$

where $\mathcal{A}$ including color SH, opacity α, rotation $\mathbf{r}$ and scale $\mathbf{s}$. Note that the shared information and the newly introduced network are used only during training. The Gaussian attributes can be directly used for inference rendering without compromising the real-time advantage.

4.2. Residual Vectors in GS branch

Residual Feature. The feature at the same spatial point yields distinct information in NeRF and GS branches: NeRF uses the feature to predict density and color, while GS requires an additional derivation of the geometric properties (rotation and scale). Therefore, NeRF-optimized features $\mathbf{f}$ may lack adaptability for predicting Gaussian properties. To address this, we optimize a residual feature vector $\Delta \mathbf{f}$ for each Gaussian to capture information discrepancies in the shared features. This refined feature vector maintains consistency with the NeRF branch features while enabling individual Gaussians to fine-tune specific information, thus enhancing the rendering quality of the GS branch.

Residual Position. Due to potential NeRF fitting errors and differing spatial perception caused by the GS branch, the Gaussian position $\mathbf{p}$ derived from NeRF branch initialization may not fully suit the GS branch. Therefore, in addition to the residual feature, we optimize a residual position $\Delta \mathbf{p}$ for each Gaussian to capture subtle spatial adjustments.

After introducing the discrepancy modeling, Gaussian attributes can be derived based on the adjusted position and feature vectors.

$$\mathcal{A} = F_{gs}(\mathbf{p} + \Delta \mathbf{p}, \mathbf{f} + \Delta \mathbf{f}), \tag{7}$$

where $\Delta \mathbf{p}$ and $\Delta \mathbf{f}$ are modeled as trainable parameters as shown in Fig. 2.

4.3. Joint Optimization in Dual-branch

GS-Rays. NeRF requires dense sampling and network queries, which preclude rendering an entire image in a single pass like in 3DGS. To synchronize optimization, we propose rendering NeRF using only partial rays in each iteration. We select rays that connect Gaussian positions with the high opacity in the current view frustum space to the camera origin, which we refer to as GS-Rays: $\mathcal{R}_{gs}$. For the k-th training view, its GS-Rays are determined by the corresponding camera origin $\mathbf{o}^k$ and the ray directions $\mathbf{d}_i^k$.

$$\mathcal{R}_{gs}^k = \{\mathbf{o}^k, \mathbf{d}_i^k\}, \ \mathbf{d}_i^k = \mathbf{p}_i^k - \mathbf{o}^k, \tag{8}$$

where $\mathbf{p}_i^k$ is visible Gaussian positions with the high opacity in the k-th view. This design ensures that the sampling points in the NeRF branch are distributed as closely as possible to the Gaussian spheres, thereby aligning the scene perception and enhancing the effectiveness of the shared information across different branches, as well as providing the necessary data for subsequent joint optimization.

Growing and Pruning Operation. In the original 3DGS, Gaussian growth occurs via gradient evaluation, which restricts gradient awareness to regions containing Gaussian spheres, potentially overlooking important blank areas. Inspired by Point-NeRF [61], we leverage NeRF spatial continuity to address this limitation. Specifically, we evaluate the

opacity at sampling points in the NeRF branch as follows.

$$\alpha_{nerf} = 1 - \exp(-\sigma_i \delta_i). \qquad (9)$$

New Gaussian spheres are then added at points with high opacity and far from existing Gaussian spheres. We regulate NeRF-driven growing to achieve an optimal balance between the number of Gaussian spheres and scene representation accuracy, which mitigates the 3DGS limitation of localized growth perception.

For pruning, we adopt the original 3DGS strategy, relying solely on GS branch information without NeRF assistance. This is because the pruning regions already include Gaussian-perceptible areas.

Loss Design. During joint training, we design loss functions for single-branch optimization and dual-branch collaboration. For the NeRF branch, we use an L1 norm loss $\mathcal{L}_{\text{nerf}}^{\text{rgb}}$ for rendered RGB values and an entropy loss $\mathcal{L}_{\text{nerf}}^{\text{en}}$ [28] for density, which promotes a more concentrated distribution pattern by discouraging density dispersion.

$$\mathcal{L}_{\text{nerf}} = \mathcal{L}_{\text{nerf}}^{\text{rgb}} + \lambda_{\text{en}} \mathcal{L}_{\text{nerf}}^{\text{en}}. \qquad (10)$$

For the GS branch, we use an L1 norm loss $\mathcal{L}_{\text{gs}}^{\text{rgb}}$ and SSIM loss $\mathcal{L}_{\text{gs}}^{\text{SSIM}}$ for rendered images, along with a volume regularization $\mathcal{L}_{\text{gs}}^{\text{vol}}$ [35] to minimize Gaussian sphere overlap.

$$\mathcal{L}_{\text{gs}} = \mathcal{L}_{\text{gs}}^{rgb} + \lambda_{\text{SSIM}} \mathcal{L}_{\text{gs}}^{\text{SSIM}} + \lambda_{\text{vol}} \mathcal{L}_{\text{gs}}^{\text{vol}}. \qquad (11)$$

For dual-branch collaborative loss, we use L1 norm $\mathcal{L}_{\text{joint}}^{\text{rgb}}$ to constrain the rendered pixel values along GS-Rays in the NeRF branch with corresponding GS branch rendered pixels. Additionally, we align the Gaussian opacity (α in Eq. 3) with corresponding NeRF opacities (α_{nerf} in Eq. 9) by L1 norm loss $\mathcal{L}_{\text{joint}}^{\text{op}}$. Residual features and position residuals are further constrained by L2 norm regularization, denoted as $\mathcal{L}_{\text{reg}}^{\text{fea}}$ and $\mathcal{L}_{\text{reg}}^{\text{pos}}$, to encourage NeRF and GS branches to learn common spatial properties while providing mutual regularization against overfitting. The overall loss function during joint optimization is as follows:

$$\begin{aligned} \mathcal{L}_{\text{total}} = & \mathcal{L}_{\text{gs}} + \lambda_{\text{nerf}} \mathcal{L}_{\text{nerf}} + \lambda_{\text{rgb}} \mathcal{L}_{\text{joint}}^{\text{rgb}} + \\ & \lambda_{\text{op}} \mathcal{L}_{\text{joint}}^{\text{op}} + \lambda_{\text{fea}} \mathcal{L}_{\text{reg}}^{\text{fea}} + \lambda_{\text{pos}} \mathcal{L}_{\text{reg}}^{\text{pos}}. \qquad (12) \end{aligned}$$

5. Experiments

5.1. Implementation Details

Training and Optimization Details. Following the original 3DGS method configurations, we implement NeRF-GS in PyTorch. For the Hash network, we set 16 different level grids, each outputting 2-dimensional features, yielding a 32-dimensional feature vector for a spatial point. During the NeRF branch pre-training, each batch contains 8,192 rays and is trained for 10 epochs. For real-world datasets,

we initialize using 1,000,000 points sampled at an 8:2 ratio from edge rays and random rays, while Blender datasets are initialized with 100,000 points. To enhance efficiency during joint training, the NeRF branch renders 4,096 rays sampled from GS-Rays. We set λ_{en}, λ_{SSIM}, and λ_{vol} to 1e-4, 0.2, and 1e-3, respectively, while λ_{nerf}, λ_{rgb}, λ_{op}, λ_{fea}, and λ_{pos} are set to 0.1, 0.05, 1e-3, 1e-4, and 1e-4 respectively. NeRF-assisted Gaussian growth occurs every 100 iterations, with a maximum of 200 new additions. Joint training iterates 30k for full-view datasets and 8k for sparse-view scenes. All experiments are conducted on an NVIDIA A100 GPU.

Datasets and Metrics. We report experimental results on real-world datasets, including Mip-NeRF360 (all 9 scenes) [5], Tanks&Temples [29] DeepBlending [23], and the Blender dataset [41]. Evaluation metrics include PSNR, SSIM [59], and LPIPS [68]. Additionally, we compare metrics for training time (minutes), storage size (MB), and rendering speed (FPS) to assess the model's compactness and efficiency.

Baselines. Our method is focused on enhancing GS branch performance, so we primarily compare it with 3DGS [27] and its variants, including C3DGS [44], Scaffold-GS [35], Mip3DGS [65], and 2DGS [24]. We also compare methods incorporating NeRF properties such as Hash-GS [8], and VDGS [36], as well as SplatFields [40], which is specifically designed for sparse-view scenes.

5.2. Comparison

We conduct extensive quantitative and qualitative comparisons with state-of-the-art methods on both full and sparse datasets. As our primary focus is on the impact of NeRF integration on GS performance, all results, unless otherwise noted, use the GS branch as the final output of NeRF-GS.

Full View Scene. We optimize NeRF-GS using the default full training data on multiple benchmark datasets. Comparative results are shown in Table 1, where our approach significantly outperforms the vanilla 3DGS model and other state-of-the-art methods across PSNR, SSIM, and LPIPS metrics. Qualitative experiments in Fig. 3 demonstrate our method's superior capability in capturing high-frequency textures and fine geometric details while better reflecting lighting conditions. Notably, compared to other methods that incorporate NeRF-like concepts, such as VDGS and Hash-GS, NeRF-GS achieves even more substantial improvements. This indicates that our dual-branch joint optimization framework is more effective than simple NeRF initialization or directly adapting NeRF implicit concepts, validating NeRF-GS as a robust framework for integrating diverse 3D representation approaches.

Sparse View Scene. Through shared spatial positions and corresponding encoded features, different branches within NeRF-GS can more comprehensively perceive and learn

Table 1. **Quantitative comparison on real-world datasets.** Colors denote the 1st , 2nd , and 3rd best-performing model.

| Method	DeepBlending			Mip-NeRF360			Tanks&Temples		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
INGP [42]	23.62	0.797	0.423	26.43	0.725	0.339	21.72	0.723	0.330
3DGS [27]	29.42	0.899	0.247	27.49	0.813	0.222	23.69	0.844	0.178
C3DGS [44]	29.79	0.901	0.258	27.08	0.798	0.247	23.32	0.831	0.201
Scaffold-GS [35]	30.21	0.906	0.254	27.5	0.806	0.252	23.96	0.853	0.177
Hash-GS [8]	29.98	0.902	0.269	27.53	0.807	0.238	24.04	0.846	0.187
VDGS [36]	29.54	0.906	0.243	27.64	0.813	0.220	24.02	0.851	0.176
Ours	30.70	0.912	0.237	28.32	0.817	0.210	24.44	0.860	0.161
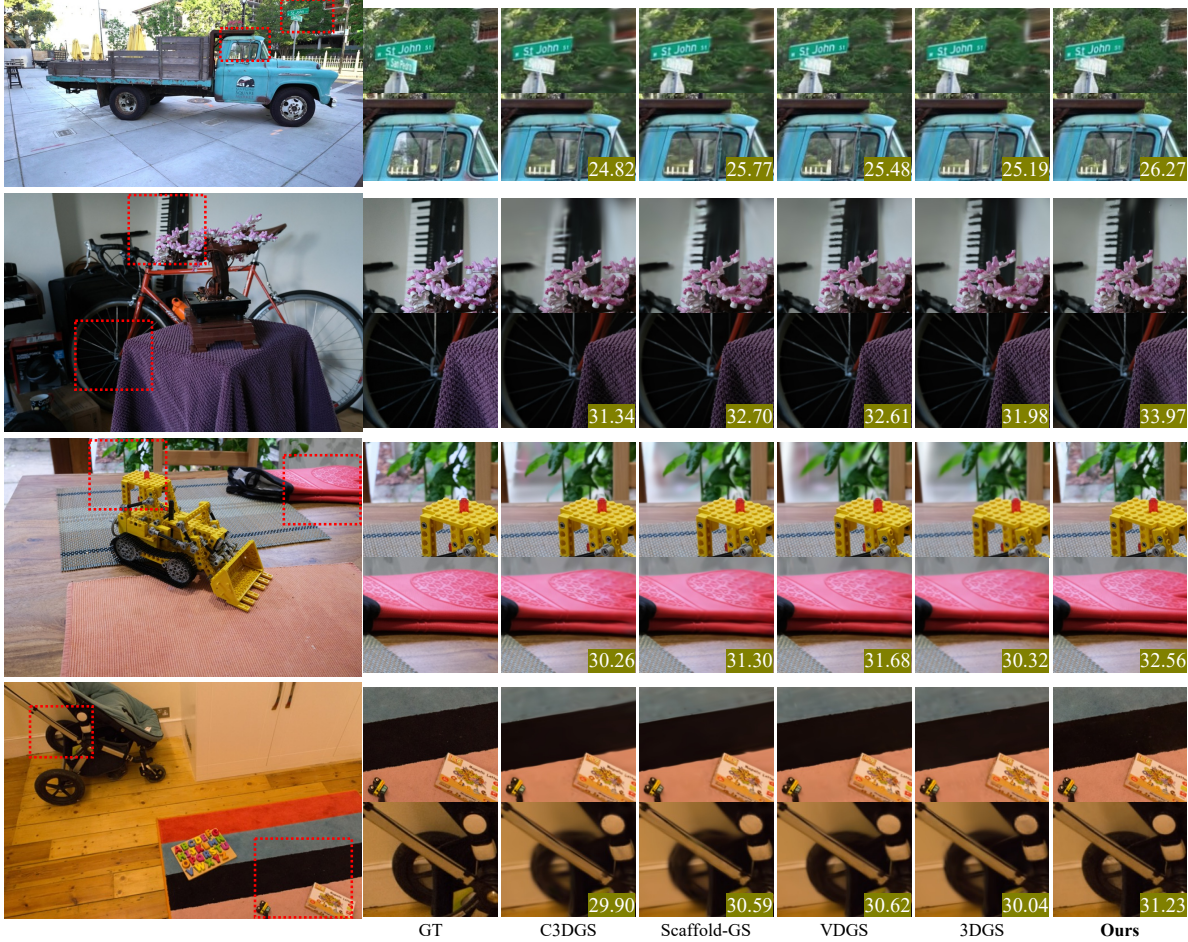
Figure 3. **Qualitative comparison on real-world datasets**. The numbers indicate the PSNR. Our method demonstrates a significant advantage over 3DGS and its variants, achieving a more faithful representation of scene details.

from limited 3D scene information. Additionally, the collaborative optimization between NeRF and GS branches, facilitated by this shared information, creates mutual constraints and regularization effects, mitigating overfitting, which is crucial for modeling scenes under sparse views. To validate this, we perform sparse-view comparisons with baseline methods, as shown in Table 2. Across various sparsity levels, NeRF-GS consistently surpasses corresponding baselines. Remarkably, NeRF-GS achieves performance comparable to or even surpassing the SplatField method, which is specifically designed for sparse-view settings. Fig. 4 also provides a qualitative illustration of these improvements. Furthermore, it can be observed that the performance gap between NeRF-GS and baseline methods in sparse views is more pronounced than in full views, affirming the effectiveness of our method's regularization. We

Figure 4. **Qualitative comparison under 12 input views on the Blender dataset**. The numbers indicate the PSNR.

Table 2. **Quantitative comparison of using different numbers of input views on Blender dataset**. Our NeRF-GS maintains high performance when the scene input views are reduced.

| Method	Full views		12 views		8 views	
	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
SparseNeRF [57]	32.46	0.957	22.92	0.875	22.20	0.861
INGP [42]	33.18	0.960	22.68	0.875	21.87	0.860
3DGS [27]	33.32	0.970	25.29	0.900	22.93	0.866
Mip3DGS [65]	33.36	0.969	24.86	0.898	22.37	0.862
Scaffold-GS [35]	33.41	0.966	23.82	0.874	21.53	0.836
2DGS [24]	33.07	0.964	25.62	0.911	23.04	0.877
Hash-GS [8]	33.24	0.967	25.36	0.909	23.14	0.879
VDGS [36]	33.37	0.969	24.77	0.898	22.88	0.872
SplatFields [40]	33.25	0.966	25.80	0.911	23.98	0.889
Ours	33.71	0.970	26.34	0.912	23.92	0.881
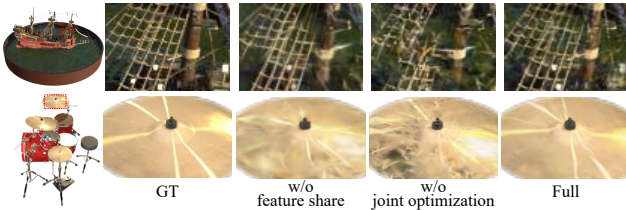
Figure 5. **Impact of feature share and joint optimization on sparse view scenes**. These two key designs enable mutual regularization constraints between NeRF and GS branches, significantly improving the visual quality of NeRF-GS in sparse views.

Table 3. **Comparison of model efficiency with 3DGS**. We report the FPS, model size (MB), training time (minutes) and PSNR. The $3DGS_L$ denotes longer iterative training (50k) for 3DGS.

Method	DeepBlending				Mip-NeRF360			
	FPS↑	Size↓	Time↓	PSNR↑	FPS↑	Size↓	Time↓	PSNR↑
3DGS	105	672	**36.1**	29.42	101	729	**41.5**	27.49
$3DGS_L$	104	678	55.7	29.50	102	733	67.9	27.57
Ours	122	526	51.7	**30.70**	134	564	60.3	**28.32**

further conduct relevant experiments in the next subsection.

Figure 6. **Visualization of position residuals**. The points represent the initial Gaussian positions, with the top 20% of points having the largest optimized residuals highlighted in red. We compare this with the results obtained by fixed Gaussian positions during training, demonstrating the importance of the residual vectors for personalized adaptation in the GS branch.

5.3. Qualitative Analysis of NeRF-GS

Regularization Effect. By introducing the spatial continuity of NeRF, NeRF-GS establishes self-correlation across different Gaussian spheres in 3DGS. Additionally, feature sharing, cross-branch loss constraints, and joint optimization enable mutual regularization between NeRF and 3DGS. In Fig. 5, we illustrate the critical role of NeRF-GS in preventing overfitting to the scene. When associations between two branches are directly removed, such as feature sharing, loss constraints during joint training, etc., the NeRF-GS shows large visual quality degradation.

Visualization of Discrepancy. Errors introduced during NeRF pre-training and inherent disparities between NeRF and 3DGS can impede the GS branch's ability to effectively model a 3D scene from NeRF-shared information. NeRF-GS addresses this challenge through a residual mechanism. An example is shown in Fig. 6, incorporating positional residuals allows the GS branch to adjust Gaussian positions, avoiding artifacts that could arise from only adjusting Gaussian shapes under fixed positions.

Model Efficiency. While NeRF-GS bridges two distinct 3D representation models, it maintains their independence. Post-joint training, each branch can retain only its effec-

Table 4. **Ablation of different components in NeRF-GS on Tank&Temples and DeepBlending datasets**. Numbers represent the PSNR metric. See Section 5.4 for discussion.

	Tanks&Temples			DeepBlending		
	Truck	Train	Avg	Drjohnson	Playroom	Avg
	Ablation of sharing mechanisms					
w/o Edge-based Init	24.06	21.17	22.61	28.65	29.8	29.8
w/o Feature Share	25.74	22.12	23.93	29.54	30.91	30.22
	Ablation of residual vectors					
w/o Residual Feature	25.88	22.31	24.09	29.76	30.84	30.3
w/o Residual Position	25.97	22.35	24.16	29.89	31.01	30.45
	Ablation of joint optimization					
w/o $\mathcal{L}_{joint}^{fea}$	26.16	22.51	24.33	30.05	30.88	30.46
w/o $\mathcal{L}_{joint}^{pos}$	26.09	22.46	24.27	30.1	31.03	30.56
w/o $\mathcal{L}_{joint}^{op}$	26.3	22.4	24.35	30.02	31.17	30.60
w/o $\mathcal{L}_{joint}^{rgb}$	26.14	22.48	24.31	30.21	30.97	30.59
w/o GS-Rays	25.82	22.26	24.04	29.85	30.6	30.22
Full	26.27	22.61	**24.44**	30.17	31.23	**30.7**

tive components, preserving the original single-branch inference speed. As shown in Table 3, our method maintains GS real-time rendering capabilities while requiring less storage than the original 3DGS approach. This is because our initialization and Gaussian growing strategies reduce Gaussian spheres. For example, on the DeepBlending dataset, vanilla 3DGS uses 2,461,023 Gaussians, while ours uses only 1,926,336. We also compare it with an extended-training version of $3DGS_L$, showing NeRF-GS outperforms 3DGS even with similar training time. This suggests that integrating the NeRF branch is a worthwhile trade-off despite the increase in training time.

5.4. Ablation Studies

Impact of Sharing Mechanisms. In NeRF-GS, information exchange manifests through spatial co-utilization and feature sharing. We propose a scene-edge-based initialization scheme as in Eq. 5 and compare it with the alternative initialization from SFM, denoted as 'w/o Edge-based Init'. Moreover, to examine the effect of feature sharing, we directly train the GS branch with learnable feature parameters, remarked as 'w/o Feature Share'. The ablation results in Table 4 indicate that our proposed initialization significantly outperforms the alternatives. Likewise, the feature-sharing scheme across NeRF and GS exhibits irreplaceable positive impacts on the full scene.

Impact of Residual Vectors. As discussed in Sec 4.2, the GS branch needs to derive different geometric information from NeRF's shared features and initial points, suggesting the need for differentiated information encoding. To achieve this, we introduce residual strategies for both features and Gaussian positions. Removing these terms results

in significant performance degradation, as shown in Table 4. This indicates that, in addition to information sharing, enabling each branch to learn adapted and differentiated information is also critical. In contrast, previous methods such as Scaffold-GS, Hash-GS and VDGS that merely incorporate NeRF characteristics overlooked this distinction, thereby offering limited performance improvement.

Impact of Joint Optimization Strategy. Our joint optimization process incorporates several key components to ensure efficient and effective training between the NeRF and GS branches. The GS-Rays strategy directs NeRF to focus on areas that are essential for the GS branch during rendering, effectively enabling efficient information exchange and mutual enhancement between branches. The term 'w/o GS-Rays' in Table 4 shows performance decline when replacing GS-Rays with an equal number of random rays. We also evaluate the effectiveness of newly introduced loss terms, as indicated in Table 4. Removing mutual constraints between branch outputs leads to performance degradation. Furthermore, applying regularization constraints to shared information to prevent excessive branch discrepancy enhances model performance.

6. Limitations

Although NeRF-GS fundamentally turns NeRF and 3DGS from competitors into collaborators and achieves superior performance, it also increases method complexity and computational overhead. Certain components within the two full pipelines may be redundant when combined. Developing a more compact and streamlined integration strategy could enhance our framework's applicability and improve its interpretability in future research.

7. Conclusion

In this study, we introduce NeRF-GS, a novel framework that combines implicit neural radiance fields with Gaussian splatting. Its core innovation lies in dual-branch collaborative design, comprising three key components: shared information in positional spaces and features, residual vectors to model inherent inter-branch differences, and joint optimization via GS-Rays alignment of intermediate results and rendering outputs, as well as adaptive Gaussian controls. These strategies effectively address several limitations of 3DGS, including initialization dependency, limited spatial awareness, insufficient Gaussian sphere correlation, and overfitting in sparse-view scenes. Experimental results demonstrate that NeRF-GS achieves state-of-the-art performance, offering new insights into the fusion of NeRF and 3DGS as an efficient hybrid approach for 3D scene representation.

Acknowledgments

References

[1] Muhammad Salman Ali, Maryam Qamar, Sung-Ho Bae, and Enzo Tartaglione. Trimming the fat: Efficient compression of 3d gaussian splats through pruning. *arXiv preprint arXiv:2406.18214*, 2024. 2

[2] Matan Atzmon, Niv Haim, Lior Yariv, Ofer Israelov, Haggai Maron, and Yaron Lipman. Controlling neural level sets. *Advances in Neural Information Processing Systems(NIPS)*, 2019. 2

[3] Milena T Bagdasarian, Paul Knoll, Florian Barthel, Anna Hilsmann, Peter Eisert, and Wieland Morgenstern. 3dgs. zip: A survey on 3d gaussian splatting compression methods. *arXiv preprint arXiv:2407.09510*, 2024. 2

[4] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5855–5864, 2021. 2

[5] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5470–5479, 2022. 5

[6] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European conference on computer vision*, pages 333–350. Springer, 2022. 1

[7] Guikun Chen and Wenguan Wang. A survey on 3d gaussian splatting. *arXiv preprint arXiv:2401.03890*, 2024. 1

[8] Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. Hac: Hash-grid assisted context for 3d gaussian splatting compression. In *European Conference on Computer Vision*, pages 422–438. Springer, 2025. 1, 2, 5, 6, 7

[9] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. In *European Conference on Computer Vision*, pages 370–386. Springer, 2025. 1

[10] Devikalyan Das, Christopher Wewer, Raza Yunus, Eddy Ilg, and Jan Eric Lenssen. Neural parametric gaussians for monocular non-rigid object reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10715–10725, 2024. 2

[11] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. *arXiv preprint arXiv:2311.17245*, 2023. 2

[12] Zhiwen Fan, Wenyan Cong, Kairun Wen, Kevin Wang, Jian Zhang, Xinghao Ding, Danfei Xu, Boris Ivanovic, Marco Pavone, Georgios Pavlakos, et al. Instantsplat: Unbounded sparse-view pose-free gaussian splatting in 40 seconds. *arXiv preprint arXiv:2403.20309*, 2, 2024. 1

[13] Shuangkang Fang, Yufeng Wang, Yi Yang, Yi-Hsuan Tsai, Wenrui Ding, Shuchang Zhou, and Ming-Hsuan Yang. Editing 3d scenes via text prompts without retraining. *arXiv e-prints*, pages arXiv–2309, 2023. 1

[14] Shuangkang Fang, Yufeng Wang, Yi Yang, Weixin Xu, Heng Wang, Wenrui Ding, and Shuchang Zhou. Pvd-al: Progressive volume distillation with active learning for efficient conversion between different nerf architectures. *arXiv preprint arXiv:2304.04012*, 2023. 1, 2

[15] Shuangkang Fang, Weixin Xu, Heng Wang, Yi Yang, Yufeng Wang, and Shuchang Zhou. One is all: Bridging the gap between neural radiance fields architectures with progressive volume distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 597–605, 2023. 2

[16] Shuangkang Fang, Dacheng Qi, Weixin Xu, Yufeng Wang, Zehao Zhang, Xiaorong Zhang, Huayu Zhang, Zeqi Shao, and Wenrui Ding. Efficient implicit sdf and color reconstruction via shared feature field. In *Proceedings of the Asian Conference on Computer Vision*, pages 3499–3516, 2024. 1

[17] Shuangkang Fang, Yufeng Wang, Yi-Hsuan Tsai, Yi Yang, Wenrui Ding, Shuchang Zhou, and Ming-Hsuan Yang. Chat-edit-3d: Interactive 3d scene editing via text prompts. *arXiv preprint arXiv:2407.06842*, 2024. 1

[18] Yalda Foroutan, Daniel Rebain, Kwang Moo Yi, and Andrea Tagliasacchi. Does gaussian splatting need sfm initialization? *arXiv preprint arXiv:2404.12547*, 2024. 1, 2, 4

[19] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5501–5510, 2022. 1, 2

[20] Yang Fu, Sifei Liu, Amey Kulkarni, Jan Kautz, Alexei A. Efros, and Xiaolong Wang. Colmap-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20796–20805, 2024. 1

[21] Jian Gao, Chun Gu, Youtian Lin, Hao Zhu, Xun Cao, Li Zhang, and Yao Yao. Relightable 3d gaussian: Real-time point cloud relighting with brdf decomposition and ray tracing. *arXiv preprint arXiv:2311.16043*, 2023. 2

[22] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 14346–14355, 2021. 1

[23] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (ToG)*, 37(6):1–15, 2018. 5

[24] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. *arXiv preprint arXiv:2403.17888*, 2024. 2, 5, 7

[25] Yingwenqi Jiang, Jiadong Tu, Yuan Liu, Xifeng Gao, Xiaoxiao Long, Wenping Wang, and Yuexin Ma. Gaussian-shader: 3d gaussian splatting with shading functions for reflective surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5322–5332, 2024.

[26] Jaewoo Jung, Jisang Han, Honggyu An, Jiwon Kang, Seonghoon Park, and Seungryong Kim. Relaxing accurate initialization constraint for 3d gaussian splatting. *arXiv preprint arXiv:2403.09413*, 2024. 2

[27] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4):1–14, 2023. 1, 2, 4, 5, 6, 7

[28] Mijeong Kim, Seonguk Seo, and Bohyung Han. Infonerf: Ray entropy minimization for few-shot neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12912–12921, 2022. 5

[29] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36 (4):1–13, 2017. 5

[30] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. *Advances in Neural Information Processing Systems*, 35:23311–23330, 2022. 2

[31] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3d gaussian representation for radiance field. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21719–21728, 2024. 2

[32] Zhihao Liang, Qi Zhang, Wenbo Hu, Ying Feng, Lei Zhu, and Kui Jia. Analytic-splatting: Anti-aliased 3d gaussian splatting via analytic integration. *arXiv preprint arXiv:2403.11056*, 2024. 2

[33] David B Lindell, Julien NP Martel, and Gordon Wetzstein. Autoint: Automatic integration for fast neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14556–14565, 2021. 2

[34] Xiangrui Liu, Xinju Wu, Pingping Zhang, Shiqi Wang, Zhu Li, and Sam Kwong. Compgs: Efficient 3d scene representation via compressed gaussian splatting. *arXiv preprint arXiv:2404.09458*, 2024. 2

[35] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20654–20664, 2024. 1, 2, 5, 6, 7

[36] Dawid Malarz, Weronika Smolak, Jacek Tabor, Sławomir Tadeja, and Przemysław Spurek. Gaussian splatting with nerf-based color and opacity. *arXiv preprint arXiv:2312.13729*, 2023. 1, 2, 5, 6, 7

[37] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR*, 2021. 1

[38] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition(CVPR)*, 2019. 2

[39] Mateusz Michalkiewicz, Jhony K Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Implicit surface representations as layers in neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision(ICCV)*, 2019. 2

[40] Marko Mihajlovic, Sergey Prokudin, Siyu Tang, Robert Maier, Federica Bogo, Tony Tung, and Edmond Boyer. Splatfields: Neural gaussian splats for sparse 3d and 4d reconstruction. In *European Conference on Computer Vision*, pages 313–332. Springer, 2025. 1, 2, 5, 7

[41] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *Proceedings of the European Conference on Computer Vision(ECCV)*, 2020. 1, 2, 5

[42] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, 2022. 1, 2, 4, 6, 7

[43] KL Navaneet, Kossar Pourahmadi Meibodi, Soroush Abbasi Koohpayegani, and Hamed Pirsiavash. Compact3d: Compressing gaussian splat radiance field models with vector quantization. *arXiv preprint arXiv:2311.18159*, 2023. 2

[44] Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3d gaussian splatting for accelerated novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10349–10358, 2024. 2, 5, 6

[45] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision(ICCV)*, 2019. 2

[46] Michael Niemeyer, Fabian Manhardt, Marie-Julie Rakotosaona, Michael Oechsle, Daniel Duckworth, Rama Gosula, Keisuke Tateno, John Bates, Dominik Kaeser, and Federico Tombari. Radsplat: Radiance field-informed gaussian splatting for robust real-time rendering with 900+ fps. *arXiv preprint arXiv:2403.13806*, 2024. 1, 2, 4

[47] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *Proceedings of the IEEE/CVF International Conference on Computer Vision(ICCV)*, 2019. 2

[48] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision(ICCV)*, 2021. 2

[49] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation.

In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition(CVPR)*, 2019. 2

[50] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *Proceedings of the European Conference on Computer Vision(ECCV)*, 2020. 2

[51] Quentin Picard, Stephane Chevobbe, Mehdi Darouich, and Jean-Yves Didier. A survey on real-time 3d scene reconstruction with slam methods in embedded systems. *arXiv preprint arXiv:2309.05349*, 2023. 1

[52] Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, and Andrea Tagliasacchi. Derf: Decomposed radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14153–14161, 2021. 2

[53] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 14335–14345, 2021. 1, 2

[54] Taha Samavati and Mohsen Soryani. Deep learning-based 3d reconstruction: a survey. *Artificial Intelligence Review*, 56(9):9175–9219, 2023. 1

[55] Xiaowei Song, Jv Zheng, Shiran Yuan, Huan-ang Gao, Jingwei Zhao, Xiang He, Weihao Gu, and Hao Zhao. Sags: Scale-adaptive gaussian splatting for training-free anti-aliasing. *arXiv preprint arXiv:2403.19615*, 2024. 2

[56] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5459–5469, 2022. 2

[57] Guangcong Wang, Zhaoxi Chen, Chen Change Loy, and Ziwei Liu. Sparsenerf: Distilling depth ranking for few-shot novel view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9065–9076, 2023. 7

[58] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*, 2021. 1

[59] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 5

[60] Dejia Xu, Ye Yuan, Morteza Mardani, Sifei Liu, Jiaming Song, Zhangyang Wang, and Arash Vahdat. Agg: Amortized generative 3d gaussians for single image to 3d. *arXiv preprint arXiv:2401.04099*, 2024. 2

[61] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Pointnerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5438–5448, 2022. 4

[62] Zhiwen Yan, Weng Fei Low, Yu Chen, and Gim Hee Lee. Multi-scale 3d gaussian splatting for anti-aliased rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20923–20931, 2024. 2

[63] Chen Yang, Sikuang Li, Jiemin Fang, Ruofan Liang, Lingxi Xie, Xiaopeng Zhang, Wei Shen, and Qi Tian. Gaussianobject: Just taking four images to get a high-quality 3d object with gaussian splatting. *arXiv preprint arXiv:2402.10259*, 2024. 2

[64] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5752–5761, 2021. 1, 2

[65] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19447–19456, 2024. 1, 5, 7

[66] Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Soundstream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:495–507, 2021. 2

[67] Jian Zhang, Yuanqing Zhang, Huan Fu, Xiaowei Zhou, Bowen Cai, Jinchi Huang, Rongfei Jia, Binqiang Zhao, and Xing Tang. Ray priors through reprojection: Improving neural radiance fields for novel view extrapolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18376–18386, 2022. 2

[68] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 5

[69] Zhaoliang Zhang, Tianchen Song, Yongjae Lee, Li Yang, Cheng Peng, Rama Chellappa, and Deliang Fan. Lp-3dgs: Learning to prune 3d gaussian splatting. *arXiv preprint arXiv:2405.18784*, 2024. 2

[70] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa volume splatting. In *Proceedings Visualization, 2001. VIS'01.*, pages 29–538. IEEE, 2001. 3